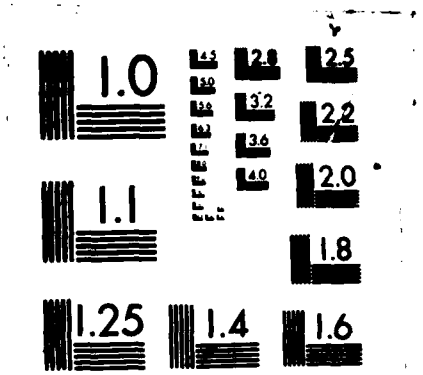
AD-A183 668 ADA (TRADEMARK) COMPILER VALIDATION SUMMARY REPORT 1/1
DIGITAL EQUIPMENT CORP. (U) FEDERAL SOFTWARE MANAGEMENT
SUPPORT CENTER FALLS CHURCH VA 06 SEP 85

UNCLASSIFIED F/G 12/5 NL



(2)

Ada* COMPILER VALIDATION SUMMARY REPORT:

Digital Equipment Corporation
VAX Ada Compiler
Version 1.1
VAX 8600, VAX-11/785, VAX-11/780,
VAX-11/782, VAX-11/750, VAX-11/730,
MicroVAX I & II, VAXstation I & II
using VAX/VMS and MicroVMS Version 4.2
and VAXELN Version 2.

September 6, 1985

Prepared by:

Federal Software Management Support Center
Office of Software Development
and Information Technology
Two Skyline Place, Suite 1100
5203 Leesburg Pike
Falls Church, VA 22041-3467

Prepared for:

Digital Equipment Corporation
110 Spit Brook Road
Nashua, NH 03062

Ada Joint Program Office
1211 Fern St., C-107
Arlington, Va 22202

* Ada is a registered trademark of the U.S. Government,
(Ada Joint Program Office).

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETEING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A165 662

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Ada Compiler Validation Summary Report:
Digital Equipment Corp., VAX Ada Compiler Version 1.1
VAX 8600, VAX-11/785, VAX-11/780, VAX-11/782, VAX-11/750, VAX-11/730, MicroVAX I&II, VAX station I&II

5. TYPE OF REPORT & PERIOD COVERED
6 SEPT 1985 to 6 SEPT 1986

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Federal Software Management Support Center

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION AND ADDRESS
Federal Software Management Support Center, Office of Software Development and Information Technology, Two Skyline Place, Suite 1100, 5203 Leesburg Pike , Falls Church, VA 22041-3467

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Ada Joint Program Office
United States Department of Defense
Washington, DC 20301-3081ASD/SIOL

12. REPORT DATE
6 SEPT 1985

13. NUMBER OF PAGES
22

14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office)
Federal Software Management Support Center

15. SECURITY CLASS (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20. If different from Report)

UNCLASSIFIED

18. SUPPLEMENTARY NOTES

19. KEYWORDS (Continue on reverse side if necessary and identify by block number)

Ada Programming language, Ada Compiler Validation Summary Report, Ada Compiler Validation Capability, ACVC, Validation Testing, Ada Validation Office, AVO, Ada Validation Facility, AVF, ANSI/MIL-STD-1815A, Ada Joint Program Office, AJPO

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Attached.

ABSTRACT

The purpose of this Validation Summary Report is to present the results and conclusions of performing standardized tests on the Digital Equipment Corporation VAX Ada Compiler. On-site testing was performed during 12-16 August 1985 at Digital Equipment Corporation, Nashua, New Hampshire, by an Ada Validation Facility (AVF), the Federal Software Management Support Center (FSMSC), according to current Ada Validation Office (AVO) policies and procedures.

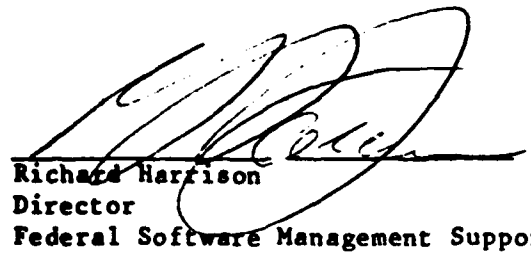
The DEC VAX Ada Compiler is hosted on those systems listed in the table below. The hosts, as well as selected VAX/VMS and VAXELN based systems served as the target systems. The suite of tests known as the Ada Compiler Validation Capability (ACVC), Version 1.6, was used. The ACVC is used to validate conformance of the compiler to ANSI/MIL-STD-1815A (1983) [Ada]. This standard is described in the ANSI Ada Reference Manual, February 17, 1983. Not all tests in the ACVC test suite are applicable to this specific implementation. Also, known test errors in Version 1.6 are present in some tests; these tests were withdrawn. The purpose of the testing is to ensure that the compiler properly implements legal language constructs and that it identifies, rejects from processing, and labels illegal constructs.

The Digital Equipment Corporation (DEC) Compiler VAX Ada, Version 1.1, using VAX/VMS 4.2, MicroVMS(*) 4.2, and VAXELN 2.0 was tested with version 1.6 of the ACVC validation tests. Version 1.6 of the test suite contains 2162 tests, of which 56 were withdrawn and 48 were inapplicable. All of the 2058 remaining tests were passed.

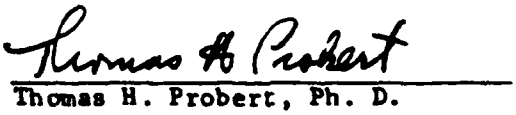
The following table represents the validated target/host relationships:

CPU TYPE	HOST OPERATING SYSTEM	TARGET OPERATING SYSTEMS
VAX 8600	VAX/VMS	VAX/VMS
VAX-11/785	VAX/VMS	VAX/VMS
VAX-11/782	VAX/VMS	VAX/VMS
VAX-11/780	VAX/VMS	VAX/VMS
VAX-11/750	VAX/VMS	VAX/VMS
VAX-11/730	VAX/VMS	VAX/VMS
MicroVAX I&II(*)	MicroVMS	MicroVMS VAXELN
VAXstation I&II(*)	MicroVMS	MicroVMS

(*) = MicroVMS is a repackaged variant of VMS for the MicroVAX I&II and the VAXstation I&II systems with limited disk space. It is not a subset and provides the full power and functionality of VMS. Throughout the remainder of this document, it will not be necessary to distinguish VMS and MicroVMS. We shall use VMS to refer to them both for easier reading. The reader can assume MicroVMS is present whenever the MicroVax or VAXstation is referenced.



Richard Harrison
Director
Federal Software Management Support Center



Thomas H. Probert, Ph. D.
Institute for Defense Analyses

Virginia Castor
Acting Director
Ada Joint Program Office

ABSTRACT

The purpose of this Validation Summary Report is to present the results and conclusions of performing standardized tests on the Digital Equipment Corporation VAX Ada Compiler. On-site testing was performed during 12-16 August 1985 at Digital Equipment Corporation, Nashua, New Hampshire, by an Ada Validation Facility (AVF), the Federal Software Management Support Center (FSMSC), according to current Ada Validation Office (AVO) policies and procedures.

The DEC VAX Ada Compiler is hosted on those systems listed in the table below. The hosts, as well as selected VAX/VMS and VAXELN based systems served as the target systems. The suite of tests known as the Ada Compiler Validation Capability (ACVC), Version 1.6, was used. The ACVC is used to validate conformance of the compiler to ANSI/MIL-STD-1815A (1983) [Ada]. This standard is described in the ANSI Ada Reference Manual, February 17, 1983. Not all tests in the ACVC test suite are applicable to this specific implementation. Also, known test errors in Version 1.6 are present in some tests; these tests were withdrawn. The purpose of the testing is to ensure that the compiler properly implements legal language constructs and that it identifies, rejects from processing, and labels illegal constructs.

The Digital Equipment Corporation (DEC) Compiler VAX Ada, Version 1.1, using VAX/VMS 4.2, MicroVMS(*) 4.2, and VAXELN 2.0 was tested with version 1.6 of the ACVC validation tests. Version 1.6 of the test suite contains 2162 tests, of which 56 were withdrawn and 48 were inapplicable. All of the 2058 remaining tests were passed.

The following table represents the validated target/host relationships:

CPU TYPE	HOST OPERATING SYSTEM	TARGET OPERATING SYSTEMS
VAX 8600	VAX/VMS	VAX/VMS
VAX-11/785	VAX/VMS	VAX/VMS
VAX-11/782	VAX/VMS	VAX/VMS
VAX-11/780	VAX/VMS	VAX/VMS
VAX-11/750	VAX/VMS	VAX/VMS
VAX-11/730	VAX/VMS	VAX/VMS
MicroVAX I&II(*)	MicroVMS	MicroVMS VAXELN
VAXstation I&II(*)	MicroVMS	MicroVMS

(*) = MicroVMS is a repackaged variant of VMS for the MicroVAX I&II and the VAXstation I&II systems with limited disk space. It is not a subset and provides the full power and functionality of VMS. Throughout the remainder of this document, it will not be necessary to distinguish VMS and MicroVMS. We shall use VMS to refer to them both for easier reading. The reader can assume MicroVMS is present whenever the MicroVax or VAXstation is referenced.

TABLE OF CONTENTS

1. Introduction

1.1 Purpose of the Validation Summary Report

This report describes the results of the validation testing for the compiler designated as VAX Ada, Version 1.1 using the following configurations:

Host Machines;	VAX 8600, VAX-11/785, VAX-11/782, VAX-11/780, VAX-11/750, VAX-11/730, MicroVAX I&II(*), VAXstation I&II(*).
Operating System;	MicroVMS 4.2 for machines marked with asterisk, VMS 4.2 for all others.
Host Disk System;	RP06, RA81, RC25
Target Machines;	VAX 8600, VAX-11/785, VAX-11/782, VAX-11/780, VAX-11/750, VAX-11/730, MicroVAX I&II, VAXstation I&II.
Operating System;	MicroVMS 4.2 or VAXELN for the MicroVAX I&II; MicroVMS 4.2 for the VAXstation I&II and VMS 4.2 for all other machines.
Language Version;	ANSI/MIL-STD-1815A (1983) [Ada]
Translator Name;	VAX Ada
Validation Test Version:	1.6

Testing of this compiler was conducted by the Federal Software Management Support Center under the supervision of the Ada Validation Office (AVO), at the direction of the Ada Joint Program Office. Testing was conducted from 12-16 August, 1985 at Digital Equipment Corporation, Nashua, NH, in accordance with AVO policies and procedures.

The purpose of this report is to document the results of the testing performed on the compiler, and in particular, to:

1) identify any language constructs supported by the compiler that do not conform to the Ada standard;

2) identify any unsupported language constructs required by the Ada standard; and

3) describe implementation-dependent behavior allowed by the standard.

1.2 Validation Overview

MicroVAX hardware implements a subset of the VAX instruction set found on the other members of the VAX series; however, all "missing" instructions are transparently implemented by software emulation under the MicroVMS variant of the VMS operating system as well as under VAXELN. MicroVMS and VAXELN provide a user mode instruction set execution environment identical to that of all other processors in the series.

VAXstation I&II are MicroVAX I&II systems, respectively. The VAXstations differ from the MicroVAX I&II only by the incorporation of a sophisticated display and related graphics support for the primary terminal device.

The VAX family includes multiple hardware/software implementations of the same instruction-set architecture. Digital Equipment Corporation maintains a VAX Architecture Management group which develops processes and tools to assure that each new VAX implementation conforms to the VAX Architecture Standard. In addition, a separate Software Quality Management Group is responsible for certifying that VAX software products work correctly on all new VAX implementations. All processors of the VAX family together with the VMS operating system provide an identical user mode instruction-set execution environment and, according to the vendor, need not be distinguished for purposes of validation. Similarly, all VAX family processors supported as VAXELN Toolkit targets provide an identical user mode instruction-set execution environment.

The VAX Ada produces equivalent results on all members of the VAX family under the VMS operating system. Similarly, VAX Ada and the VAXELN Ada run-time library produce equivalent results on those members of the VAX family under VAXELN as indicated in the abstract table.

The members of Digital's VAX family are the VAX 8600, VAX-11/785, VAX-11/780, VAX-11/782, (the VAX-11/782 is a duplex multiprocessor configuration using two VAX-11/780 processors), VAX-11/750, VAX-11/730, MicroVAX I&II, and VAXstation I&II.

1.3 Host to Target Relationship Table

The following table represents the validated target/host relationships:

CPU TYPE	HOST OPERATING SYSTEM	TARGET OPERATING SYSTEMS	
VAX 8600	VAX/VMS	VAX/VMS	
VAX-11/785	VAX/VMS	VAX/VMS	
VAX-11/782	VAX/VMS	VAX/VMS	
VAX-11/780	VAX/VMS	VAX/VMS	
VAX-11/750	VAX/VMS	VAX/VMS	
VAX-11/730	VAX/VMS	VAX/VMS	
MicroVAX I&II(*)	MicroVMS	MicroVMS	VAXELN
VAXstation I&II(*)	MicroVMS	MicroVMS	

(*) = MicroVMS is a repackaged variant of VMS for the MicroVAX I&II and the VAXstation I&II systems with limited disk space. It is not a subset and provides the full power and functionality of VMS. Throughout the remainder of this document, it will not be necessary to distinguish VMS and MicroVMS. We shall use VMS to refer to them both for easier reading. The reader can assume MicroVMS is present whenever the MicroVax or VAXstation is referenced.

1.4 Use of the Validation Summary Report

The Ada Validation Office may make full and free public disclosure of this report in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation apply only to the computers, operating systems, and compiler version identified in this report.

The Ada Compiler Validation Capability is used to determine insofar as is practical, the degree to which the subject compiler conforms to the Ada standard. Thus, this report is necessarily discretionary and judgemental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in this report are accurate or complete, nor that the subject compiler has no other nonconformances to the Ada standard. This report is not meant to be used for the purpose of publicizing the findings summarized herein.

Any questions regarding this report or the validation tests should be sent to the Ada Validation Office at:

Federal Software Management Support Center
5203 Leesburg Pike
Suite 1100
Falls Church, Virginia 22041

1.5 References

Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A-1983, February 1983.

Ada Validation Organization: Policies and Procedures, Mitre Corporation, June 1982, PB 83-110601.

Ada Compiler Validation Implementers' Guide, SofTech, Inc., October 1980.

The Ada Compiler Validation Capability, Computer, Vol. 14, No. 6, June 1981.

Using the ACVC Tests, SofTech, Inc., November 1981.

Ada Compiler Validation Plans and Procedures, SofTech, Inc., November 1981.

1.6 Definitions of Terms

Class A tests are passed if no errors are detected at compile time. Although these tests are constructed to be executable, no checks can be performed at run-time to see if the test objective has been met; this distinguishes Class A from Class C tests. For example, a Class A test might check that keywords of other languages (other than those already reserved in Ada) are not treated as reserved words by an Ada implementation.

Class B tests are illegal programs. They are passed if all the errors they contain are detected at compile time (or link time) and no legal statements are considered illegal by the compiler.

Class L tests consist of illegal programs whose errors cannot be detected until link time. They are passed if errors are detected prior to beginning execution of the main program.

Class C tests consist of executable self-checking programs. They are passed if they complete execution and do not report failure.

Class D tests are capacity tests. Since there are no firm criteria for the number of identifiers permitted in a compilation, number of units in a library, etc., a compiler may refuse to compile a class D test. However, if such a test is successfully compiled, it should execute without reporting a failure.

Class E tests provide information about an implementation's interpretation of the Standard. Each test has its own pass/fail criterion.

ACVC:	Acronym for the Ada Compiler Validation Capability.
AVO:	The Ada Validation Office. In the context of this report the AVO is responsible for directing compiler validation.
CHECK or CHECKTEST:	An automated tool that produces summary test results by reading compiler output in a spool file.
CUSTOMER:	The agency requesting the validation (Digital Equipment Corporation).

FSTC:	Federal Software Management Support Center. In the context of this report the FSMSC conducts Ada validations under contract to the AVO as a satellite facility.
HOST:	The computer on which the compiler executes (VAX 8600, VAX-11/785, VAX-11/782, VAX-11/780, VAX-11/750, VAX-11/730, MicroVAX I & II, VAXstations I & II)
IG:	ACVC Implementers' Guide.
RM:	The Ada Language Reference Manual.
STANDARD:	The standard for the Ada language, ANSI/MIL-STD-1815A (1983) [Ada].
SUBSET TESTS:	A grouping of ACVC tests selected by the FSMSC. Each chapter in the ACVC is represented in the subset by between 4 to 7 tests. The subset is used for statistical sampling of the various host and target hardware configurations.
TARGET:	The computers for which the compiler generates object code (VAX 8600, VAX-11/785, VAX-11/782, VAX-11/780, VAX-11/750, VAX-11/730, VAX-11/725, MicroVAX I&II, VAXstation I&II).
VALIDATION:	The process of testing a compilation system to certify that it conforms to the standard.
VALIDATION TESTS:	The set of test programs used to detect non-conformances in compilation systems. In this report, the term will be used (unqualified) to mean the ACVC tests.

2. TEST ANALYSIS

The following table shows that the DEC VAX Ada compiler passed all applicable tests.

	A	B	C	D	E	L	Total
In Suite	61	800	1273	17	8	3	2162
Inapplicable	0	4	44	0	0	0	48
Withdrawn	0	13	43	0	0	0	56
Passed	61	783	1186	17	8	3	2058
Failed	0	0	0	0	0	0	0

48 tests in the suite were found to be inapplicable to the Digital implementation.

In addition, 56 tests were withdrawn from the test suite because they were incorrect programs.

2.1 Class A Testing

Class A tests check that legal Ada programs can be successfully compiled. These tests are executed but contain no executable self-checking capabilities. There were 61 class A test programs processed in this validation.

2.1.1 Class A Test Procedures

Each class A test was separately compiled and executed. However, the only purpose of execution is to produce a message indicating that the test passed.

2.1.2 Class A Test Results

Successful compilation and execution without any error messages indicates the tests passed. All 61 applicable tests were passed.

2.2 Class B Testing

Class B tests check the ability to recognize illegal language usage. There were 783 applicable class B tests processed.

2.2.1 Class B Test Procedures

Each Class B test was separately compiled. The resulting test compilation listings are manually examined to see whether every illegal construct in the test is detected. If some errors are not detected, a version of the program test is created that contains only undetected illegal constructs.

This revised version is recompiled and the results analyzed. If some errors are still not detected, the revision process is repeated until a revised test contains only a single previously undetected illegal construct.

A B test is considered to fail only if a version of the test containing a single illegal construct is accepted by the compiler (i.e., an illegal construct is not detected) or a version containing no errors is rejected (i.e., a legal construct is rejected).

2.2.2 Class B Test Results

There were 800 class B tests presented to the compiler. Of these tests 4 were found to be inapplicable to this implementation (see Section 4.2.7); 13 tests were found to be incorrect (i.e., a conforming compiler would have failed each of these tests). All 783 remaining class B tests were passed.

Because all errors were not detected when compiling the original tests, 2 tests were modified by removing the detected errors; the modified tests were then resubmitted to see if the remaining errors would be detected. These tests were:

B97101A-AB.ADA and B97101E-AB.ADA.

All illegal constructs were detected except in the tests that were withdrawn because of errors in them (see Section 4.2.6).

2.3 Class C Testing

Class C tests check that legal Ada programs are correctly compiled and executed by an implementation. There were 1186 class C tests processed in this validation attempt.

2.3.1 Class C Test Procedures

Each Class C test is separately compiled and executed. The tests are self-checking and produce pass/fail messages. Any 'failed' tests are individually checked to see if they are correct and if they are applicable to the implementation. Any tests that are inapplicable or that do not conform to the Ada Standard are withdrawn.

2.3.2 Class C Test Results

All 1186 applicable class C tests were processed and passed.

2.4 Class D Testing

Class D tests are executable tests used to check an implementation's compilation and execution capacities. There were 17 class D tests used in this validation.

2.4.1 Class D Test Procedures

Each class D test is separately compiled and executed. The tests are self-checking and produce PASS/FAIL messages.

2.4.2 Class D Test Results

All 17 D test were passed.

2.5 Class E Testing

Class E tests are executable tests that provide information about an implementation's interpretation of the Standard in areas where the Standard permits implementations to differ. Each test has its own pass/fail criterion. There were 8 class E tests used in this validation.

2.5.1 Class E Testing Procedures

Each class E test is separately complied and executed. The tests are self-checking and produce pass/fail messages.

2.5.1 Class E Test Results

All class E tests were passed.

2.6 Class L Testing

Three Class L tests check that incomplete or illegal Ada programs involving multiple separately compiled source files are detected at link time and are not allowed to execute. There were 3 Class L test programs processed in this validation attempt.

2.6.1 Class L Test Procedures

Each Class L test is separately compiled and execution is attempted. The tests produce FAIL messages if executed. Any "failed" tests are individually checked to see if they are correct and if they are applicable to the implementation. Any tests that are inapplicable or that do not conform to the Ada standard are withdrawn.

2.6.2 Class L Test Results

Of the 3 class L tests, none was found to be inapplicable to this implementation, and none was withdrawn due to errors in the tests. All three L tests were passed.

2.7 Subset Testing

A subset of the executable ACVC tests was defined by the FSMSC and used during the this Ada validation. This subset of tests was used to test multiple configuration combinations during the pre-validation. The subset comprised of the following tests:

Chapter 2	Chapter 3	Chapter 4	Chapter 5
C23001A	C34001A	C41101D	C51002A
C24102A	C34001H	C42005A	C52001A
C26008A	A32203D	C43214A	C53005A
A29002A	C35904A	C45101A	C54A03A
	C36204A	C48003A	D55A03A
	C34002B		

Chapter 6	Chapter 7	Chapter 8	Chapter 9
C61003B	A71002A	A83A02A	C92002A
A62006D	C72001B	C84002A	C93001A
C63004A	C74203B	C85007E	C94006A
C65003A	C74209A	C86003A	A97106A
C66002A	C74409B	C87B48A	C97202A

Chapter 10	Chapter 11	Chapter 12	Chapter 14
CA1003A	CB1001A	CC1004A	AE2101A
CA2004A0M	CB2004A	CC3004A	CE2102A
CA2004A1	CB3003A	CC3408A	CE2201A
CA2004A2	CB4001A	CC3504C	CE2401E
CA2004A3			CE3102A
CA2004A4			CE3901A

*** CZ ***

CZ1101A
CZ1102A
CZ1103A
CZ1201A
CZ1201B
CZ1201C
CZ1201D

3. COMPILER ANOMALIES AND NONCONFORMANCE

There were no nonconformances to the Ada standard detected in this validation. The compiler passed all applicable correct tests.

4. ADDITIONAL INFORMATION

This section describes in more detail how the validation was conducted.

4.1 Compiler Parameters

Certain tests do not apply to all Ada compilers, e.g., compilers are not required to support several predefined floating-point types, and so tests must be selected based on the predefined types an implementation actually supports. In addition, some tests are parameterized according to the maximum length allowed by an implementation for an identifier (or other lexical element; this is also the maximum line length), the maximum floating-point precision supported, etc.. The implementation-dependent parameters used in performing this validation were:

- maximum lexical element length: 120 characters.
- maximum digits value for floating point types: 33
- SYSTEM.MIN_INT: -2147483648
- SYSTEM.MAX_INT: 2147483647
- predefined numeric types: INTEGER, FLOAT, SHORT-INTEGER, SHORT-SHORT-INTEGER, LONG-FLOAT, LONG-LONG-INTEGER
- INTEGER'FIRST: -2147483648
- INTEGER'LAST: 2147483647
- source character set: ASCII
- extended ascii chars: abcdefghijklmnopqrstuvwxyz !$%?@[/]
- non-ascii char type: (NON_NULL)
- TEXT_IO.COUNT'LAST: 2147483647
- TEXT_IO.FIELD'LAST: 2147483647
- illegal external file name1: "badcharacter*©"
- illegal external file name2: "much_too_long_name_for_a file_much_too_long_name_for_a_file"
- SYSTEM.PRIORITY'FIRST: 0
- SYSTEM.PRIORITY'LAST: 15

4.2 Testing Information

Tests were comiled/executed at digital Equipment Corporation, Nashua, NH.

4.2.1 Pre-Test Procedures

Prior to testing, appropriate values for the compiler-dependent parameters were determined. These values were used to adapt tests that depend on the values (This relates to on-site testing!). A magnetic tape containing the adapted tests (and split versions of some class B tests--see section 2.2.2) was prepared and brought to the testing site.

The Digital pre-validation consisted of the following steps:

a) The entire ACVC was processed on both the VAX-11/785 and VAX-11/782 under VMS. The results were examined and found to be correct. The results from the VAX-11/785 were used as the "test basis" against which all other results were compared to determine correctness.

b) The load module created on the VAX-11/785 in step "a" was transported to and executed on the MicroVAX I under VMS. The results were compared to the test basis and found to be correct.

c) A subset of the ACVC executable tests (see 2.7) was then compiled and executed on the VAX 8600 and on the MicroVAX II under VMS. Both results were examined and found to be correct.

d) All results were compared to the test basis; no differences were detected.

4.2.2 Control Files

Digital Equipment Corporation provided command procedures that compiled and executed tests automatically.

4.2.3 On-site Data Collection

The complete ACVC was run on both the VAX 8600 under VMS and the MicroVAX II under MicroVax. The results from the VAX 8600 were analyzed; they were determined to be correct. These results were thus used as the "test basis" against which the results from running the ACVC on other configurations would be compared. The various test results were captured on tape; the tapes were compared against the test basis with "DIF"--a difference utility. Each ACVC run can be linked twice to produce two load modules, (1) a VMS load module which executes under VMS and MicroVMS and (2) a VAXELN load module which executes on a bare machine after loading its own VAXELN operating system. Only load modules from the 8600 were executed under VAXELN.

The results on the MicroVAX II were compared against the test basis; no differences were detected.

The VAXELN-linked load module from the VAX 8600 was transported to the MicroVAX II and executed under VAXELN. The results were checked against the test basis and found to be correct.

The VMS-linked load module from the MicroVAX II under VMS was transported to the VAX-11/780 and executed under VMS. The results were checked against the test basis and found to be correct.

Additionally, the two results of re-targetting the two load modules described above were compared against each other using the DIF utility; no differences were detected.

The VMS-linked load module from the MicroVAX II was transported to, and executed on, the following configurations:

VAX-11/750
VAX-11/730
VAXstation I&II, and
MicroVAX I.

The results were individually compared against the test basis and found to be correct.

Also, the results created during pre-validation for the VAX-11/785 & VAX-11/782 were compared against the test basis and found to be correct.

4.2.4 Test Analysis Procedures

On completion of testing the base system, all results were analyzed for failed Class A, C, D, E, or L programs, and all class B compilation results were individually analyzed. Analysis procedures are described for each test class in chapter 2.

4.2.5 Timing Information

The real (i.e., wall clock) times required for compiling the non-executable tests, and compiling, linking, and running the executable tests were -

VAX 8600	3:10	Full run of validation suite under VAX/VMS - multiple batch streams run in parallel.
MicroVAX II	14:17	Full run of validation suite under MicroVMS - single batch stream.

4.2.6 Description of Errors in Withdrawn Tests

The following tests in version 1.6 of the ACVC did not conform to the ANSI Ada standard and were withdrawn for the reasons given below:

- B66001A-B: Test checks (in section G) that a parameterless function that is equivalent to an enumeration literal in the same declarative region is a redeclaration and, as such, is forbidden. According to RM 8.3(17), the explicit declaration of such a function is allowed if an enumeration literal is considered to be an implicitly declared predefined operation. The RM is not clear on this point. This issue has been referred to the Language Maintenance Committee for resolution. Since the issue cannot be resolved at this time, the test is withdrawn from Version 1.6. (Please note that this test may be considered correct and may appear in the future Versions of the ACVC, including Version 1.6.)

- BC1013A-B: The declaration of equality in lines 86-87 is illegal because the parameter type T declared in line 11 is not a limited type (LRM 6.7-4).

- C45521A, C45521B, C45521C, ... C45521Y (25 tests): Cases C and I in each of these tests define the model interval for the result too narrowly.

- C48005C-B: Lines 38 and 63 of this test should check that the value of the designated object is null.

- C64103C-B: This test should raise CONSTRAINT_ERROR during the conversion at line 179.

- C64103D-B: This test involves a CONSTRAINT_ERROR vs. NUMERIC_ERROR issue that is to be resolved by the Language Maintenance Committee.

- C64105E-AB: For case E, ensure that non-null dimensions of formal and actual parameters belong to both index subtypes (see AI-00313).

- C64105F-AB: For case E, ensure that non-null dimensions of formal and actual parameters belong to both index subtypes (See AI-00313).

- B67001A-B: Line 414 is missing the "BEGIN NULL; END;" needed to complete the block beginning at line 389 (case H).

- B67004A-B: The default name for a formal generic equality function should not be allowed to be "/=" unless an expanded name is used.

- C93005A, C93005B and C93005C: These tests contain a declaration of an integer variable whose initialization is solely for the purpose of raising an exception. Some compilers will not raise this exception due to their optimization.

- C93007B-B: This test should check for PROGRAM_ERROR rather than TASKING_ERROR (See AI-000149).

- CA1011A*-B: The test objective should be reversed to be consistent with AI-00199.

- CA1108A-B: A pragma ELABORATE is needed for OTHER_PKG at line 25.

- CA1108B-B: A pragma ELABORATE is needed for FIRST_PKG at line 39 and for LATER_PKG at line 49.

- CA2009B-B, CA2009E-B: The repetition of the main procedure after the subunit body makes the subunit body obsolete; therefore, an attempt to execute the main procedure will fail.

- CA2009F*-B: The file CA2009F2-B is missing from the test suite.

- BC3204A-B, BC3204B-B, BC3204C*-B, BC3204D-B, BC3205A-B, BC3205B-B, BC3205C-B, BC3205D*-B, BC3405B-B: Instantiations with types that have default discriminants are now legal (AI-00037).

- CE3603A-B: The last case is inconsistent with AI-00050. If string argument is null, no attempt to read is made and END_ERROR is not raised.

- CE3604A-B: Cases 5, 8, 9, and 11 are inconsistent with AI-00050. SKIP_LINE is called only if the end of the output string has not been met.

- CE3704M-B: A superfluous SKIP_LINE causes the input and output operations to be out of synchronization.

4.2.7 Description of Inapplicable Tests

B52004D and B55B09C were inapplicable because the implementation does not support LONG_INTEGER.

B86001CP and B86001CS were inapplicable because the implementation does not support SHORT_FLOAT and LONG_INTEGER, respectively.

C34001E and C34001F were inapplicable because the implementation does not have predefined types LONG-INTEGER and SHORT_FLOAT, respectively.

C35702A was inapplicable because the implementation does not have a predefined type SHORT_FLOAT.

C24113H through C24113Y, were inapplicable because the implementation does not support source lines longer than 120 characters. (18)

C55B07A inapplicable because the implementation does not have a predefined type LONG_INTEGER.

C86001F was inapplicable because the compilation of the user supplied package SYSTEM makes the body of TEXT_IO obsolete.

C96005B - This test checks to find a difference between DURATION'BASE'FIRST and DURATION'FIRST. If no difference exists (as is the case in the implementation) the test is inapplicable.

CC3407A, CC3407D, CC3407E, and CC3407F are inapplicable because NUMERIC_ERROR is raised in calculating the size of objects of a type, as allowed by the LRM.

CE2102D, CE2102E, and CE2102F are inapplicable because the implementation does support files of mode IN_FILE, OUT_FILE, and INOUT_FILE, respectively.

CE2102G is inapplicable because the implementation does support RESET and DELETE.

CE2107B, CE2107C, CE2107D, CD2107E, CE2110B, and CE3114B are inapplicable because the implementation does not support associating two internal files with the same external file for writing.

CE2111D is inapplicable because VAX Ada does not allow the creation of a file of mode IN_FILE.

CE3111B, CE3111C, CE3111D, and CE3111E were inapplicable because the implementation does not support associating two internal files with the same external files with the same external name when one internal file has made OUT_FILE.

CE3115A was inapplicable because more than one internal file was associated with the same external file.

4.2.8 Information Derived from the Tests

Processing of the following tests indicated support as described below for a variety of implementation options examined by the tests.

- E24101A-B.TST: If a based integer literal has a value exceeding SYSTEM.MAX_INT, an implementation may either reject the compilation unit at compile time or raise NUMERIC_ERROR at run time. This test showed that NUMERIC_ERROR was raised at run time.

- B26005A.ADA: This test contains all the ASCII control characters in string literals. The system replaced the control characters other than those corresponding to format effectors with a space in the listing file. All occurrences were identified with a diagnostic message by the compiler.

- D29002K-B.ADA: This test declares 713 identifiers and was passed by the compiler.

- E36202A-B.ADA and E36202B-B.ADA: These tests declare multidimensional null BOOLEAN arrays in which LENGTH of one dimension exceeds INTEGER'LAST and SYSTEM.MAX_INT, respectively. An implementation can accept this, or it can raise NUMERIC_ERROR or STORAGE_ERROR at run time. The compiler did accept the declarations and raised NUMERIC_ERROR during execution.

- D4A002A-AB.ADA, D4A002B.ADA, D4A004A.ADA and D4A004B.ADA: These tests contain universal integer calculations requiring 32 and 64 bits of accuracy, i.e., values that exceed SYSTEM.MAX_INT are used. An implementation is allowed to reject programs requiring such calculations. The compiler passed all four tests.

- E43211B-B.ADA: If a bound in a non-null range of a non-null aggregate does not belong to an index subtype, then all choices may or may not be evaluated before CONSTRAINT_ERROR is raised. The compiler evaluates all choices before CONSTRAINT_ERROR is raised.

- E43212B-B.ADA: This test examines whether or not all choices are evaluated before subaggregates are checked for identical bounds. The compiler evaluates all subaggregates first.

- E52103Y-B.ADA, C52104X-B.ADA, C52104Y-B.ADA: These tests declare BOOLEAN arrays with INTEGER'LAST+3 components. An implementation may raise NUMERIC_ERROR at the type declaration or STORAGE_ERROR when array objects of these types are declared, or it may accept the type and object declarations. The compiler did not raise NUMERIC_ERROR for null array with one dimension of length greater than INTEGER'LAST in E52103Y-B.

- A series of tests (D55A03*-AB.ADA) checks to see what level of loop nesting is allowed by an implementation. Tests containing up to 65 nested loops passed without exceeding the implementation's capacity.

- D56001B-AB.ADA contains blocks nested 65 levels deep. This test was passed.

- C94004A-B.ADA: This test checks to see what happens when a library unit initiates a task and a main program terminates without ensuring that the library unit's task is terminated. An implementation is allowed to terminate the library unit task or it is allowed to leave the task in execution. This test showed that such library tasks continue to execute.

- BC3204C*-B.ADA and BC3205D*-B.ADA: These tests contain a separately compiled generic declaration, some instantiations, and a body. An implementation must reject either the instantiations or the body. The compiler generated errors when compiling the generic package body.

- CE2106A-B.DEP and CE3110A-B.DEP: These tests confirm that dynamic creation and deletion of files is supported.

- CE2107*.DEP: These tests showed that more than one internal file may be associated with the same external file for reading.

- EE3102C-B.ADA: This test confirmed that an Ada program can open an existing file in OUT_FILE mode, and can create an existing file in either OUT_FILE or IN_FILE mode.

- CE3111A-B.DEP confirmed that two internal files may read the same external files.

- CE3111B-B.DEP and CE3111C-B.DEP showed that the compiler does not allow two internal TEXT_IO files to be associated with the same external file when one or both internal files are opened for writing.

5. SUMMARY AND CONCLUSIONS

The Ada Validation Facility (AVF) identified 2162 of the ACVC version 1.6 tests as being being potentially applicable to the validation of the Digital Equipment Corporation compiler hosted on the VAX 8600, VAX-11/785, VAX-11/782, VAX-11/780, VAX-11/750, VAX-11/730, MicroVAX I&II and the VAXstation I&II. Of these, 56 were withdrawn due to test errors, and 48 were determined to be inapplicable. The compiler passed the remaining 2058 tests.

The AVF considers these results to show acceptable compliance to the February 1983 ANSI Ada Reference Manual.

END

9-87

DTIC